**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BOARD OF PATENT APPEALS AND INTERFERENCES**

In re Application of

Gustavson et al.

Serial No.: 10,671,937 Group Art Unit: 2192

Filed: September 29, 2003 Examiner: Wei, Zheng

For: METHOD AND STRUCTURE FOR PRODUCING HIGH PERFORMANCE LINEAR ALGEBRA ROUTINES USING PRELOADING OF FLOATING POINT REGISTERS

Commissioner of Patents
Alexandria, VA 22313-1450

APPELLANTS' BRIEF ON APPEAL

Sir:

Appellants respectfully appeal the rejection of claims 1-20 in the Office Action mailed on October 16, 2007. A Notice of Appeal was timely filed on February 15, 2008.

I. REAL PARTY IN INTEREST

The real party in interest is International Business Machines Corporation, assignee of 100% interest of the above-referenced patent application.

II. RELATED APPEALS AND INTERFERENCES

There are no other appeals or interferences known to Appellants, Appellants' legal representative or Assignee which would directly affect or be directly affected by or have a bearing on the Board's decision in this appeal.

III. STATUS OF CLAIMS

Claims 1-20 are all the claims presently pending in the application.

Claims 4, 5, 9, 10, 15, 16, and 18-20 stand rejected under 35 U.S.C. § 112, second paragraph, as being indefinite. Claims 1, 2, 17, and 20 stand rejected under 35 U.S.C. § 103(a) as unpatentable over U.S. Patent No. 5,438,669 to Nakazawa et al., further in view of US Patent 6,115,730 to Dhablania et al. Claims 3-16, 18, and 19 stand rejected under 35 U.S.C. § 103(a) as allegedly unpatentable over Nakazawa/Dhablania, further in view of Dongarra, et al., "A Set of Level 3 Basic Linear Algebra Subprograms."

Although Appellants do not completely understand the concerns underlying the 35 U.S.C. § 112, second paragraph, rejection, this rejection is not being appealed, since it is believed that it can be readily resolved upon remand. Moreover, the claim revisions in the version of the claims entered in the Appendix may have appropriately addressed the Examiner's concerns.

The prior art rejections are both being appealed.

IV. STATUS OF AMENDMENTS

An Amendment Under 37 CFR §1.116 was filed on December 17, 2007. In the Advisory Action mailed on January 30, 2008, the Examiner indicated that the arguments in the Amendment Under 37 CFR §1.116 were not persuasive and that the rejections were maintained for all claims. The Examiner also indicated that this Amendment would be entered into the record for purpose of Appeal. Therefore, the claims in the appendix represent the version of the claims in this Amendment Under 37 CFR §1.116, filed on December 17, 2007.

V. SUMMARY OF CLAIMED SUBJECT MATTER

Bases in the specification for the claims:

1. (Rejected) A software method of improving at least one of efficiency and speed (lines 13-15 of page 6) in executing a linear algebra subroutine on a computer (lines 15-21 of page 6) having a floating point unit (FPU) and a load/store unit (LSU) capable of overlapping loading data and processing said data by the FPU, said method comprising:

for an execution code controlling operation of said linear algebra subroutine execution, overlapping by preloading data into floating point registers (FRegs) of said FPU (lines 5-8 of page 13), said overlapping causing data to arrive into said FRegs to be timely executed by the FPU operations of said linear algebra subroutine on said FPU (lines 19-21 of page 12).

6. (Rejected) An apparatus, comprising:

a memory (304, Figure 3) to store matrix data to be used for processing in a linear algebra program (lines 15-21 of page 6);

an L1 cache (401, Figure 4) to receive data from said memory;

a floating point unit (FPU) (302, Figure 3) to perform said processing; and

a load/store unit (LSU) (301, Figure 3) to load data to be processed by said FPU, said LSU loading said data into a plurality of floating point registers (FRegs) (303, Figure 3),

wherein said data processing overlaps said data loading (lines 5-8 of page 13) such that matrix data is preloaded into said FRegs from said L1 cache prior to being required by said FPU (lines 19-21 of page 12).

12. (Rejected) A computer-readable medium (500, Figure 5) tangibly embodying a program of machine-readable instructions executable by a digital processing apparatus to perform a method of improving at least one of speed and efficiency (lines 13-15 of page 6)

in executing a linear algebra subroutine on a computer (lines 15-21 of page 6) having a floating point unit (FPU) and a load/store unit (LSU) capable of overlapping loading data and processing said data, said method comprising:

for an execution code controlling operation of said linear algebra subroutine execution, overlapping by preloading data into a floating point registers (FRegs) of said FPU (lines 5-8 of page 13), said overlapping causing data from an L1 cache to arrive into said FRegs, to be timely executed by FPU operations of said linear algebra subroutine on said FPU (lines 19-21 of page 12).

17. (Rejected) A method of providing a service involving at least one of solving and applying a scientific/engineering problem, said method comprising at least one of:

using a linear algebra software package that computes one or more matrix subroutines (lines 15-21 of page 6, lines 1-4 of page 20), wherein said linear algebra software package generates an execution code controlling a load/store unit loading data into a floating point registers (FRegs) for a floating point unit (FPU) performing a linear algebra subroutine execution, said FPU capable of overlapping loading data and performing said linear algebra subroutine processing (lines 5-8 of page 13), such that, for an execution code controlling operation of said FPU, said overlapping causes a preloading of data from an L1 cache into said FRegs (lines 19-21 of page 12);

providing a consultation for purpose of solving a scientific/engineering problem using said linear algebra software package (lines 4-7 of page 20);

transmitting a result of said linear algebra software package on at least one of a network, a signal-bearing medium containing machine-readable data representing said result, and a printed version representing said result (lines 7-9 of page 20); and

receiving a result of said linear algebra software package on at least one of a network, a signal-bearing medium containing machine-readable data representing said result, and a printed version representing said result (lines 7-9 of page 20).

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Appellant presents the following two grounds for review by the Board of Patent Appeals and Interferences:

GROUND 1: Claims 1, 2, 17, and 20 stand rejected under 35 U.S.C. § 103(a) as unpatentable over U.S. Patent No. 5,438,669 to Nakazawa et al., further in view of US Patent 6,115,730 to Dhablania et al.

GROUND 2: Claims 3-16, 18, and 19 stand rejected under 35 U.S.C. § 103(a) as allegedly unpatentable over Nakazawa/Dhablania, further in view of Dongarra, et al., "A Set of Level 3 Basic Linear Algebra Subprograms."

VII. ARGUMENTS

GROUND 1: The rejection of claims 1, 2, 17, and 20 under 35 U.S.C. § 103(a) as unpatentable over U.S. Patent No. 5,438,669 to Nakazawa et al., further in view of US Patent 6,115,730 to Dhablania et al.

The Examiner alleges that Nakazawa, when modified by Dhablania, renders obvious the present invention described by claims 1, 2, (17), and 20, and, when further modified by Dongarra, renders obvious claims 3-16, 18, and 19.

Appellants submit, however, that there are elements of the claimed invention which are neither taught nor suggested by Nakazawa.

As explained in the second sentence (e.g., in column 1 at lines 12-17), Nakazawa addresses a processing method in a computer architecture in which a cache is not so effective. In this architecture, a large number of data registers accessible to main memory are used. To overcome this design deficiency, Nakazawa introduces preloading of data from main memory directly into these data registers.

In contrast, in the present invention, the L1 cache is used for the matrix data transfer between main memory and the FPUs, so that Nakazawa clearly teaches against the approach of the present invention. That is, the present invention clearly addresses the data preloading into the FPUs using the cache and does so using a software mechanism that does not require the hardware registers of the Nakazawa computer architecture, an entirely different approach from that used by primary reference Nakazawa.

The latest rejection fails to address this difference discussed above and, instead, simply introduces secondary reference Dhablania.

In summary, whatever other similarities might be shared, the method in Nakazawa provides a hardware solution to get matrix data from memory to the processor and will work only for the 1995 hardware described therein. In contrast, the present invention provides a general software solution for matrix multiplication, as further enhanced if used in combination with concepts of one or more of the six other co-pending applications.

Nakazawa teaches clearly that matrix data using his processing unit cannot be efficiently retrieved from memory using typical cache architecture. In contrast, the present invention provides basic ground rules for preloading in the context of matrix multiplication kernels. This is novel and not obvious from Nakazawa. That is, relative to independent claim 1 (as well as remaining independent claims), Nakazawa would work only on the (now-obsolete) 1995 hardware, whereas the present invention is much faster and works with standard cache-based machines.

In the latest rejection, the Examiner adds the following rejection component, in an attempt to answer the plain meaning of the claim language that Appellants noted as being unsatisfied by Nakazawa:

"... *Nakazawa does not explicitly disclose the detailed method about overlapping by preloading data. However, Dhablania in the same analogous art of reloadable floating point unit, discloses a software method of improving at least one of efficiency and speed in executing a linear algebra subroutine on a computer having a floating point unit (FPU) and a load/store unit (LSU) capable of overlapping loading data and processing of said data by the FPU, said method comprising:*

For an execution code controlling operation of said linear algebra subroutine execution, overlapping by preloading data into a floating point registers (Fregs) of said FPU, said overlapping causing data to arrive into said Fregs to be timely executed by the FPU operations of said linear algebra subroutine on said FPU (see for example, Fig. 4a, 4b and related text; also see col. 1, section "Summary of the invention", "ability to initiate a next instruction held in a 4-deep instruction queue before a prior instruction has finished"; col. 4, lines 21-26, "The FPU 70 includes a load/store stage with 4-deep load and store queues")[.]

Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to use the method disclosed by Nakazawa and Dhablania to improve the performance of an FPU by providing it with preload registers which enable initiation of a next instruction held in a[n] instruction queue as suggested by Dhablania (see for example, col. 1, Summary of the invention)[.]"

In response, Appellants respectfully submits that the above-recited addition to the rejection is clearly engaging in improper hindsight, since there is no reasonable rationale in the latest rejection in accordance with any of the seven rationales proposed by the USPTO in the Federal Register Notices dated October 10, 2007, as published in the aftermath of the US Supreme Court holding in *KSR*. Instead, the rejection merely makes a conclusory statement of the purported benefit of modifying primary reference Nakazawa by newly-cited secondary reference Dhablania. That is, the rationale of record is merely a circular argument that indicates that at least a variation of elements needed to satisfy the description of the claimed invention can be found in various art references.

Appellants believe that the problem with the evaluation of record is that the Examiner's initial burden is not satisfied until there is a reasonable rationale articulated on the record. As the US Supreme Court stated in the *KSR* holding: "*'[R]ejections on obviousness cannot be sustained by mere conclusory statements; instead, there must be some articulated reasoning with some rational underpinning to support the legal conclusion of obviousness.'*"

Therefore, in the prior art evaluation of the present invention, since primary reference Nakazawa already has an FPU and a computer architecture based on a large number of data registers, the Examiner has the initial burden to revise this underlying computer architecture in Nakazawa with the FPU of newly-cited Dhablania, using one of the seven rationales recently identified by the USPTO in the Federal Register. Without articulating one of these seven rationales, the rejection currently of record fails to establish a prima facie rejection under the *KSR* holding.

More significantly, even if the FPU of newly-cited Dhablania were to be introduced into Nakazawa, the problem being addressed by the present invention would not be solved by this FPU, since, as Appellants pointed out in the previous response and have repeated above and, as clearly described in recently-added dependent claim 20, the present invention is about a penalty between the FPU registers and the L1 cache of newer computer architectures not represented by either Nakazawa or Dhablania, since Nakazawa

is clearly based on a large number of data registers, not L1 cache, as Appellants have pointed out.

Newly-cited Dhablania is a patent about hardware. It describes processor technology of the year 1997. In contrast, the present invention relates to processor technology that was introduced during 2004 and later. The present invention overcomes the lack of hardware feature that was not present in 1997. Simply put, Dhablania has little, if anything at all, to do with the present invention.

In the present invention, we need to address the case where the elements in L1 are not contiguous. Dhablania does not apply in the cases that the present invention addresses. Referring to the bullet on page 8 of the latest Office Action, the loads of 2004 and beyond that the present invention is using are load multiple or SIMD load k where $k > 1$. k is small, typically 2 or 4. These loads happen in a single cycle.

In contrast, for newly-cited Dhablania, even in the good case where the data elements are stride 1, the cost of loading is more than one cycle. In the development of the present invention, the inventors had to live with the lack of a hardware feature. They were able to find a software solution for a large class of matrix algebra subroutines.

One of the aspects that the present invention is addressing is the improvement that secondary reference Dongarra called upon manufacturers to provide, but fails itself to provide a solution.

That is, the present invention improves the performance of the routines that Dongarra has proposed to be needed. The improvement of the present invention is beyond what one would get if Nakazawa were combined with newly-cited Dhablania. The present invention discloses a general technique of preloading that is applicable to several of the routines that Dongarra discloses. Appellants note that Dongarra is only asking for innovation to be supplied by computer vendors and does not provide the solution.

The present invention supplies such an innovation. It is not found in Nakazawa or in Dhablania, or in their combination.

In the Advisory Action mailed on January 30, 2008, the Examiner states: "*At page 9, the applicant points the difference between [the] present invention and prior art*

Nakazawa, wherein Nakazawa provides a hardware solution and the present invention provides a general software solution. Therefore, the Applicant contends that the latest rejection fails to address this difference. However, the Examiner's position is that Nakazawa provides a data processing method for [preloading] data to floating point [registers] (see for example, col. 7, lines 61-62, "provide a data processing method and data processor") [emphasis added]. It is obvious that such method can be implemented either by a hardware implementation or a software solution to perform the same function for preloading data to floating point registers. Furthermore, it should be noted that the data processor itself cannot perform and/or operate without any operating instructions, i.e. software."

In response to the above-recited paragraph from the Advisory Action, Appellants respectfully traverse the Examiner's assertion that being able to perform a function in either hardware or software somehow renders obvious a solution achieved in one or the other. More specifically, there is not suggestion in Nakazawa that its hardware solution can similarly be executed in software. Therefore, there is clearly no enablement in Nakazawa for a corresponding software solution, let alone a software solution that would completely eliminate the hardware that defines the invention of this prior art reference.

In short, Appellants submit that the software solution of the present invention, assuming that the functions achieved are equivalent, is clearly a non-obvious alternative to the hardware solution offered by Nakazawa, and the rejection of record is merely begging the question in asserting that obviousness results because the Examiner considers that the same function could be done either by hardware or by software.

Clearly, there are different principles of operation involved in achieving a function by using software versus hardware. Appellants submit that the present invention, even assuming equivalence of function, achieves the function in a much more efficient and cost effective manner, since it eliminates the hardware circuits of Nakazawa and can be used in any computer without having to add the hardware described in that reference.

Therefore, Appellants submit that the software method of the present invention is not at all equivalent to the hardware method of Nakazawa and the solution of the present invention is patentably distinquished from the hardware method of Nakazawa.

In the Advisory Action, the Examiner also alleges: "*At page 10-13, the Applicant submits that there is no reasonable [rationale] in the latest rejection to combine Nakazawa and Dhablania. However, the examiner respectfully disagrees. [Nakazawa] discloses a data processing method to use preload [instructions] for reading data from a main memory [directly] to [registers], but does not explicitly [disclose] how to overlap by preloading data to cache. Dhablania discloses the detail [of the] overlapping method by preload and queue (cache) 4-deep instructions before a prior instruction has finished. Therefore, [combining] the preload method introduced by Dhablania with [Nakazawa] can further improve the performance of execution as suggested by Dhablania (see for example, col. 1, section "Summary of the Invention").*"

In response, Appellants respectfully submit that the above-recited paragraph from the Advisory Action clearly demonstrates the insidious nature of improper hindsight, when the evaluation attempts to consider techniques described in prior art references as concepts in the abstract. That is, the Examiner clearly intends to argue that one having ordinary skill in the art would be able to freely incorporate the concept of overlapping, as taught by secondary reference Dhablania, into the concept of preloading, as taught by primary reference Nakazawa, as an improvement to the technique of Nakazawa.

Appellants disagree that the rejection of record has established a reasonable rationale to modify Nakazawa using the rationale that Dhablania provides a known improvement of the technique described in Nakazawa, or that the claimed invention would result even if these two references were to be combined.

More specifically, the Examiner's reasoning has been clarified in the Advisory Action as being based upon the third rationale described in the October 10, 2007, Notices of the Federal Register:

> "*(C) Use of Known Technique To Improve Similar Devices (Methods, or Products) in the Same Way To reject a claim based on this rationale, Office personnel must resolve the Graham factual inquiries. Office personnel must then articulate the following:*

(1) a finding that the prior art contained a "base" device (method, or product) upon which the claimed invention can be seen as an "improvement;"

(2) a finding that the prior art contained a "comparable" device (method, or product that is not the same as the base device) that was improved in the same way as the claimed invention;

(3) a finding that one of ordinary skill in the art could have applied the known "improvement" technique in the same way to the "base" device (method, or product) and the results would have been predictable to one of ordinary skill in the art; and

(4) whatever additional findings based on the Graham factual inquiries may be necessary, in view of the facts of the case under consideration, to explain a conclusion of obviousness.

The rationale to support a conclusion that the claim would have been obvious is that a method of enhancing a particular class of devices (methods, or products) was made part of the ordinary capabilities of one skilled in the art based upon the teaching of such improvement in other situations. One of ordinary skill in the art would have been capable of applying this known method of enhancement to a "base" device (method, or product) in the prior art and the results would have been predictable to one of ordinary skill in the art. The Supreme Court in KSR noted that if the actual application of the technique would have been beyond the skill of one of ordinary skill in the art, then using the technique would not have been obvious. If any of these findings cannot be made, then this rationale cannot be used to support a conclusion that the claim would have been obvious to one of ordinary skill in the art."

Presumably, the Examiner intends that the hardware-based method of primary reference Nakazawa satisfies the first finding recited above and that the second finding above is satisfied by secondary reference Dhablania. However, Appellants submit that the third finding is <u>not</u> satisfied, in accordance with the characterization of these references as provided in the Advisory Action by the Examiner himself.

The third finding requires that the known improvement of the second finding be applied <u>in the same way</u> as in the claimed invention by preloading data to cache. According to the Examiner's characterization, primary reference Nakazawa preloads data from main memory directly to registers used for processing and secondary reference Dhablania provides an overlapping/preloading using cache. Therefore, Appellants submit that the Examiner's initial burden to satisfy the third finding recited above requires that

that the Examiner demonstrate that the method of Nakazawa will benefit in the same way as Dhablania was benefited.

Appellants submit that the method described in secondary reference Dhablania cannot be implemented in primary reference Nakazawa without having to defeat the purpose of the additional hardware registers initially placed in Nakazawa to overcome its own design problem.

Nor is there any evidence on record that Nakazawa would actually benefit from preloading data into cache rather than directly into the registers, as the Examiner characterizes happening in Nakazawa.

Therefore, for at least the above reasons, Appellants submit that the rejection currently of record fails to provide a reasonable rationale to modify primary reference Nakazawa to achieve the method described in the claimed invention.

Hence, turning to the clear language of the claims, in Nakazawa there is no teaching or suggestion of: "A software method of improving at least one of efficiency and speed in executing a linear algebra subroutine in a computer having a floating point unit (FPU) capable of overlapping loading data and processing of said data, said method comprising: for an execution code controlling operation of said FPU performing said linear algebra subroutine execution, overlapping by preloading data into a floating point register (FReg) of said FPU, said overlapping causing data to arrive into said FReg to be timely executed by FPU operations of said linear algebra subroutine on said FPU", as required by independent claim 1. The remaining independent claims have similar language and/or include additional limitations above those of claim 1.

Finally, it is brought to the Board's attention that the combination of Nakazawa and Dhablania would not result in the claimed invention because the preloading of the present invention relates to the newer instructions of the LSU register set not used on either of these two references. This aspect of the present invention is implied in the discussion beginning on page 12 of the disclosure but has become more significant as the newer architectures have evolved since the filing of the present application in 2003. More specifically:

(1) Floating units have instructions for moving data between the registers that hold the floating point operands, and moving/loading is overlapped with SIMD floating point operation;

(2) SIMD FMA's are balanced with SIMD loads, and data must be stride one for SIMD loads to occur;

(3) SIMD loads can always be made, however, data may be now in the wrong order in the register file; and

(4) equation (E) and (A) on page 14 of the specification is the basis of why simd loading (wrongly) can work, as follows:

i) SIMD load wrongly ahead of time: this is preloading combined with fast SIMD supply of data;

ii) use move data instructions to correct the incorrect simd loading (feature of modern attached accelerated units). Note, moving and SIMD FMA's are overlapped;

iii) incorrectly loaded data is now in the correct order for a SIMD FMA: Equation (E) and (A) give time slots to move incorrectly loaded data; and

iv) issue SIMD FMA: preloading amount is governed by first arrival, plus the move time.

For Power 3 described in the application, as filed, there was no SIMD load as SIMD vector length was 1. In that case (3) never applied. In this case iv becomes the first arrival time, which is part of the claimed invention, since the move time is zero.

Therefore, Appellant submits that there are elements of the claimed invention that are not taught or suggested by Nakazawa. Therefore, the Examiner is respectfully requested to withdraw this rejection for claims 1, 2, 17, and 20.

GROUND 2: The rejection of claims 3-16, 18, and 19 under 35 U.S.C. § 103(a) as allegedly unpatentable over Nakazawa/Dhablania, further in view of Dongarra, et al., "A Set of Level 3 Basic Linear Algebra Subprograms"

Claims 2-16, 18, and 19 stand rejected as unpatentable over Nakazawa, further in view of Dongarra. However, regardless of the propriety of combining Dongarra with Nakazawa, this secondary reference does not overcome the basic deficiency identified above for Nakazawa.

Moreover, the following comments relate to the non-obviousness of additional dependent claims over the prior art of record.

Relative to claim 4, as discussed on page 10 of the latest Office Action, Appellants again respectfully point out that use of L1 BLAS and what followed, L2 BLAS, do not work efficiently on today's cache-based architecture. The reason is that they are very slow methods. Nakawaza and Dhablania do not disclose the method of Claim 1. Also, to paraphrase Dongarra, when he asks suppliers to "build a better mouse trap", the present invention has built that better mouse trap.

Relative to claim 5, Appellants again point out that Nakazawa and, particularly Dongarra, do not use level L3 routines for factorization per se, and their methods there are very slow compared to those used in the present invention and the six co-pending applications. It was not obvious to Dongarra to use fast methods for factorization per se.

Relative to claim 18, Dongarra does not disclose any detail on how to implement fast BLAS. Dongarra is actually asking computer manufacturers to implement his methods, rather than disclosing fast methods to do this.

Finally, relative to secondary reference Dongarra, Appellants again point out that they are not attempting to patent matrix multiplication per se. Level 3 BLAS are an industry standard for matrix multiplication. Cayley defined matrix multiplication in 1854 for the first time. Dongarra merely repeats that definition in very slow implementation of matrix multiplication.

Therefore, Appellants submit that all claims are clearly patentable over Nakazawa, even if combined by Dongarra and newly-cited Dhablania, and respectfully request the Examiner to reconsider and withdraw these rejections.

For the reasons stated above, the claimed invention is fully patentable over the reference, and the Board is respectfully requested to reconsider and withdraw this rejection.

IX. CONCLUSION

In view of the foregoing, Appellants submit that claims 1-20, all the claims presently pending in the application, are clearly enabled and patentably distinct from the prior art of record and in condition for allowance. Thus, the Board is respectfully requested to remove all rejections of claims 1-20.

Please charge any deficiencies and/or credit any overpayments necessary to enter this paper to Assignee's Deposit Account number 50-0510.

Respectfully submitted,

Frederick Cooperrider

Dated: April 15, 2008

Frederick E. Cooperrider
Reg. No. 36,769

McGinn Property Law Group, PLLC
8231 Old Courthouse Road, Suite 200
Vienna, VA 22182-3817
(703) 761-4100
Customer Number: 21254

CLAIMS APPENDIX

The claims, as reflected upon entry of the Amendment Under 37 CFR §1.116 filed on December 17, 2007, are shown below:

1. (Rejected) A software method of improving at least one of efficiency and speed in executing a linear algebra subroutine on a computer having a floating point unit (FPU) and a load/store unit (LSU) capable of overlapping loading data and processing said data by the FPU, said method comprising:

for an execution code controlling operation of said linear algebra subroutine execution, overlapping by preloading data into floating point registers (FRegs) of said FPU, said overlapping causing data to arrive into said FRegs to be timely executed by the FPU operations of said linear algebra subroutine on said FPU.

2. (Rejected) The method of claim 1, wherein instructions are unrolled repeatedly until the data loading reaches a steady state in which a data loading exceeds a data consumption.

3. (Rejected) The method of claim 1, wherein said linear algebra subroutine comprises a matrix multiplication operation.

4. (Rejected) The method of claim 1, wherein said linear algebra subroutine comprises a subroutine equivalent to a LAPACK (Linear Algebra PACKage) subroutine.

5. (Rejected) The method of claim 4, wherein said LAPACK subroutine comprises a BLAS Level 3 L1 cache kernel.

6. (Rejected) An apparatus, comprising:

a memory to store matrix data to be used for processing in a linear algebra program;

an L1 cache to receive data from said memory;

a floating point unit (FPU) to perform said processing; and

a load/store unit (LSU) to load data to be processed by said FPU, said LSU loading said data into a plurality of floating point registers (FRegs),

wherein said data processing overlaps said data loading such that matrix data is preloaded into said FRegs from said L1 cache prior to being required by said FPU.

7. (Rejected) The apparatus of claim 6, wherein said preloading is achieved by unrolling a loading instruction so that a load occurs every cycle until a preload condition has been satisfied.

8. (Rejected) The apparatus of claim 6, wherein said linear algebra program comprises a matrix multiplication operation.

9. (Rejected) The apparatus of claim 6, wherein said linear algebra program comprises a subroutine equivalent to a LAPACK (Linear Algebra PACKage) subroutine.

10. (Rejected) The apparatus of claim 9, wherein said subroutine comprises a BLAS Level 3 L1 cache kernel.

11. (Rejected) The apparatus of claim 6, further comprising:

a compiler to generate an instruction for said preloading.

12. (Rejected) A computer-readable medium tangibly embodying a program of machine-readable instructions executable by a digital processing apparatus to perform a method of improving at least one of speed and efficiency in executing a linear algebra subroutine on a computer having a floating point unit (FPU) and a load/store unit (LSU) capable of overlapping loading data and processing said data, said method comprising:

for an execution code controlling operation of said linear algebra subroutine execution, overlapping by preloading data into a floating point registers (FRegs) of said FPU, said overlapping causing data from an L1 cache to arrive into said FRegs, to be timely executed by FPU operations of said linear algebra subroutine on said FPU.

13. (Rejected) The computer-readable medium of claim 12, wherein a load instruction is unrolled repeatedly until the data loading reaches a steady state in which a data loading exceeds a data consumption.

14. (Rejected) The computer-readable medium of claim 12, wherein said linear algebra program comprises a matrix multiplication operation.

15. (Rejected) The computer-readable medium of claim 12, wherein said linear algebra program comprises a subroutine equivalent to a LAPACK (Linear Algebra PACKage) subroutine.

16. (Rejected) The computer-readable medium of claim 15, wherein said subroutine comprises a BLAS Level 3 L1 cache kernel.

17. (Rejected) A method of providing a service involving at least one of solving and applying a scientific/engineering problem, said method comprising at least one of:

using a linear algebra software package that computes one or more matrix subroutines, wherein said linear algebra software package generates an execution code controlling a load/store unit loading data into a floating point registers (FRegs) for a floating point unit (FPU) performing a linear algebra subroutine execution, said FPU capable of overlapping loading data and performing said linear algebra subroutine processing, such that, for an execution code controlling operation of said FPU, said overlapping causes a preloading of data from an L1 cache into said FRegs;

providing a consultation for purpose of solving a scientific/engineering problem using said linear algebra software package;

transmitting a result of said linear algebra software package on at least one of a network, a signal-bearing medium containing machine-readable data representing said result, and a printed version representing said result; and

receiving a result of said linear algebra software package on at least one of a network, a signal-bearing medium containing machine-readable data representing said result, and a printed version representing said result.

18. (Rejected) The method of claim 17, wherein said linear algebra subroutine comprises a subroutine equivalent to a LAPACK (Linear Algebra PACKage) subroutine.

19. (Rejected) The method of claim 18, wherein said subroutine comprises a BLAS Level 3 L1 cache kernel.

20. (Rejected) The method of claim 1, wherein said FPU comprises said FRegs as interfaced with an L1 cache, said interface having a loading penalty of n cycles, said preloading eliminating this n-cycle penalty.

EVIDENCE APPENDIX

None

RELATED PROCEEDINGS APPENDIX

None